

International Journal of Wavelets, Multiresolution and Information Processing
 © World Scientific Publishing Company

How to refine polynomial functions

Henning Thielemann
Institut für Informatik
Martin-Luther-Universität Halle-Wittenberg
06110 Halle
Germany
henning.thielemann@informatik.uni-halle.de

Received 28.11.2010

Accepted 26.03.2011

Revised 13.07.2011

Extended 23.11.2011

Research on refinable functions in wavelet theory is mostly focused to localized functions. However it is known, that polynomial functions are refinable, too. In our paper we investigate on conversions between refinement masks and polynomials and their uniqueness.

Keywords: Refinable Function; Wavelet Theory; Polynomial.

AMS Subject Classification: 42C40,

1. Introduction

Refinable functions are functions that are in a sense self-similar: If you add shrunken translates of a refinable function in a weighted way, then you obtain that refinable function again. For instance, see Figure 1 for how a quadratic B-spline can be decomposed into four small B-splines and how the so called DAUBECHIES-2 generator function is decomposed into four small variants of itself.

All B-splines with successive integral nodes are refinable, but there are many more refinable functions that did not have names before the rise of the theory of refinable functions. In fact we can derive a refinable function from the weights of the linear combination in the refinement under some conditions.

Refinable functions were introduced in order to develop a theory of real wavelet functions that complements the discrete sub-band coding theory.⁶ Following the requirements of wavelet applications, existing literature on wavelets focuses on refinable functions that are \mathcal{L}_2 -integrable and thus have a well-defined FOURIER transform, are localized (finite variance) or even better of compact support. It is already known, that polynomial functions are refinable as well.² In this paper we want to explore in detail the connection between polynomials and the respective weights for refinement.

Our results can be summarized as follows:

2 Henning Thielemann

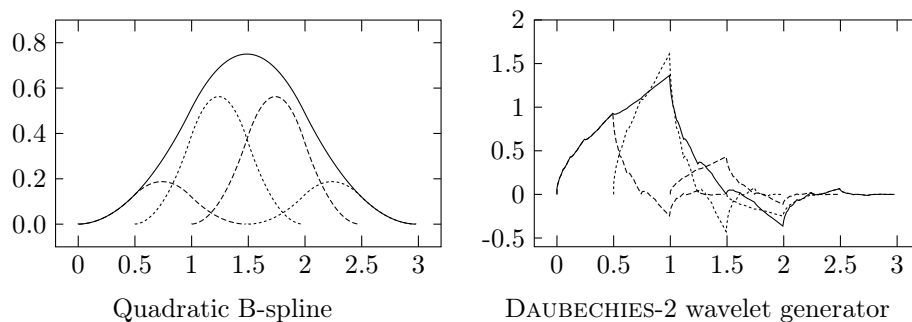


Figure 1. Refinement of a quadratic B-spline and the orthogonal DAUBECHIES-2 generator. The black line is the refinable function that is composed from shrunk, translated and weighted versions of itself, displayed with dashed lines.

- Masks that sum up to a negative power of two refine polynomials that are uniquely defined up to constant factors. Other masks are not associated with a polynomial. (Theorem 2.1)
- For every polynomial there are infinitely many refinement masks, and these refinement masks can be characterized in a simple form. (Theorem 2.2 and Theorem 2.3)
- There is a simple iterative algorithm for approximating a polynomial that is associated with a mask. (Theorem 2.4)

2. Main Work

2.1. Basics

We start with a precise definition of refinable functions.

Definition 2.1 (Refinable function). *The vector m with $m \in \mathbb{R}^{\mathbb{Z}}$ and a finite number of non-zero entries ($m \in \ell_0(\mathbb{Z})$) is called a refinement mask for the function φ if*

$$\varphi(t) = 2 \cdot \sum_{j \in \mathbb{Z}} m_j \cdot \varphi(2 \cdot t - j) \quad (2.1)$$

holds. Vice versa the function φ is called refinable with respect to the mask m .

The factor 2 before the sum is chosen, such that the following law (Lemma 2.1) about convolutions holds. Unfortunately this enforces adding or subtracting 1 here and there in some of the other theorems. There seems to be no convention that asserts overall simplicity.

Definition 2.2 (Convolution). *For sequences h and g the convolution is defined by*

$$(h * g)_k = \sum_{j \in \mathbb{Z}} h_j \cdot g_{k-j}$$

and for real functions the convolution is defined by

$$(\varphi * \psi)(t) = \int_{\mathbb{R}} \varphi(\tau) \cdot \psi(t - \tau) \, d\tau \quad .$$

Lemma 2.1. *If φ is refinable with respect to h and ψ is refinable with respect to g , then $\varphi * \psi$ is refinable with respect to $h * g$.*

For a proof see 8. For the proof of our theorems we need two further lemmas about differentiation and integration.

Lemma 2.2. *If the function φ is refinable with respect to mask m , then its derivative φ' is refinable with respect to mask $2 \cdot m$.*

Proof. Derive both sides of the refinement equation with respect to t .

$$\begin{aligned} \varphi(t) &= 2 \cdot \sum_{j \in \mathbb{Z}} m_j \cdot \varphi(2 \cdot t - j) \\ \varphi'(t) &= 2 \cdot \sum_{j \in \mathbb{Z}} m_j \cdot 2 \cdot \varphi'(2 \cdot t - j) \end{aligned} \quad \square$$

Lemma 2.3. *If the function φ is refinable with respect to mask m and there is an antiderivative Φ with $\Phi(t) = \int_0^t \varphi(\tau) \, d\tau$, then the antiderivative $(t \mapsto \Phi(t) + c)$ is refinable with respect to mask $\frac{1}{2} \cdot m$ where the constant c must be chosen as follows:*

- For $\sum_j m_j \neq 1$ it must be $c = \frac{\sum_j m_j \cdot \Phi(-j)}{1 - \sum_j m_j}$.
- For $\sum_j m_j = 1$ and $\sum_j m_j \cdot \Phi(-j) = 0$ the constant c can be chosen arbitrarily.
- For $\sum_j m_j = 1$ and $\sum_j m_j \cdot \Phi(-j) \neq 0$ there is no valid value for the constant c .

These choices for c are the only possible ones.

Proof. We start with the necessary condition; that is, given that the antiderivative $(t \mapsto \Phi(t) + c)$ is refinable with respect to mask $\frac{1}{2} \cdot m$, what are the possible

4 Henning Thielemann

integration constants?

$$\begin{aligned}
\Phi(t) + c &= 2 \cdot \sum_{j \in \mathbb{Z}} \frac{1}{2} \cdot m_j \cdot (\Phi(2 \cdot t - j) + c) \\
\Phi(t) + c \cdot (1 - \sum_{j \in \mathbb{Z}} m_j) &= \sum_{j \in \mathbb{Z}} m_j \cdot \int_0^{2 \cdot t - j} \varphi(\tau) \, d\tau \\
&= \sum_{j \in \mathbb{Z}} m_j \cdot \left(\int_{-j}^{2 \cdot t - j} \varphi(\tau) \, d\tau + \Phi(-j) \right) \\
&= \sum_{j \in \mathbb{Z}} m_j \cdot \left(2 \cdot \int_0^t \varphi(2 \cdot \tau - j) \, d\tau + \Phi(-j) \right) \\
&= \int_0^t \left(\sum_{j \in \mathbb{Z}} 2 \cdot m_j \cdot \varphi(2 \cdot \tau - j) \right) \, d\tau + \sum_{j \in \mathbb{Z}} m_j \cdot \Phi(-j) \\
&= \int_0^t \varphi(\tau) \, d\tau + \sum_{j \in \mathbb{Z}} m_j \cdot \Phi(-j) \\
c \cdot (1 - \sum_{j \in \mathbb{Z}} m_j) &= \sum_{j \in \mathbb{Z}} m_j \cdot \Phi(-j)
\end{aligned}$$

Proof of the sufficient condition: By substituting c by the admissible values we verify, that the antiderivative with that offset is actually refined by $\frac{1}{2} \cdot m$. \square

Remark 2.1. For $\sum_j m_j = 1$ and m and φ with support, that is bounded on at least one side, the second case of Lemma 2.3 applies. This means that all antiderivatives of φ irrespective of the integration constant are refinable with respect to $\frac{1}{2} \cdot m$.

Sketch of the proof: Without loss of generality let m and φ have support, that is bounded at the left. The refinement equation implies, that the bounds of their support are equal. If their support is entirely on the positive real axis, then $\forall t \leq 0 \, \Phi(t) = 0$ and further on in all summands of $\sum_j m_j \cdot \Phi(-j)$ at least one factor is zero. This implies $\sum_j m_j \cdot \Phi(-j) = 0$.

Now, when φ is refined by m and k is an integer, then φ translated by k is refinable with respect to m translated by k . This way we can reduce all m and φ with bounded support to ones with support on the positive real axis.

2.2. Conversions between polynomials and masks

If we generalize refinable functions to refinable distributions, then the DIRAC impulse is refined by the mask δ with $\delta_j = \begin{cases} 1 & : j = 0 \\ 0 & : j \neq 0 \end{cases}$ and the k -th derivative of the DIRAC impulse is refined by $2^k \cdot \delta$. Vice versa the truncated power function ($t \mapsto t_+^k$)

with $k \in \mathbb{N}$ and $t_+ = \begin{cases} t & : t \geq 0 \\ 0 & : t < 0 \end{cases}$ is refined by $2^{-k-1} \cdot \delta$. Intuitively said, truncated power functions are antiderivatives of the DIRAC impulse.

Once we are thinking about truncated power functions, we find that ordinary power functions with natural exponents are also refinable. Then it is no longer a surprise, that polynomial functions are refinable, too. For example f with $f(t) = 1 + 2 \cdot t + t^2$ is refined by the mask $\frac{1}{8} \cdot (3, -3, 1)$:

$$\begin{aligned} & 2 \cdot \frac{1}{8} \cdot (3 \cdot f(2t) - 3 \cdot f(2t-1) + 1 \cdot f(2t-2)) \\ &= \frac{1}{4} \cdot (3 \cdot (1 + 2 \cdot 2t + (2t)^2) - 3 \cdot (1 + 2 \cdot (2t-1) + (2t-1)^2) \\ &\quad + (1 + 2 \cdot (2t-2) + (2t-2)^2)) \\ &= \frac{1}{4} \cdot (3 \cdot (1 + 4t + 4t^2) - 3 \cdot 4t^2 + (1 - 4t + 4t^2)) \\ &= 1 + 2 \cdot t + t^2 \quad . \end{aligned}$$

Now that we have an example of a refinable polynomial function, we like to know how we can find a mask that refines a polynomial function. Vice versa we want to know a characterization of masks that refine polynomial functions and what polynomial functions can be refined by a given mask.

Before we start answering these questions we would like to stress the difference between a *polynomial* and a *polynomial function*.

Definition 2.3 (Polynomial and Polynomial function). *A polynomial p of degree n is a vector from $\mathbb{R}^{\{0, \dots, n\}}$. We need this for the actual computations and for performing linear algebra. A polynomial function \hat{p} is a real function. The refinement property is a property of real functions. The connection between polynomial and polynomial function is*

$$\hat{p}(t) = \sum_{k=0}^n p_k \cdot t^k \quad .$$

Our first theorem answers the question, “What polynomial can be refined by a mask?”

Theorem 2.1. *Given a mask m that sums up to 2^{-n-1} for a given natural number n , there is a polynomial p of degree n such that m refines \hat{p} . With the additional condition of the leading coefficient being 1, this polynomial is uniquely determined.*

Proof. We show this theorem by induction over n .

- Case $n = 0$

We want to show that a mask m with sum $\frac{1}{2}$ can only refine a constant polynomial. Thus we assume contrarily that m refines a polynomial with a degree d greater than zero. In the refinement relation

$$\hat{p}(t) = 2 \cdot \sum_{j \in \mathbb{Z}} m_j \cdot \hat{p}(2 \cdot t - j)$$

we only consider the leading coefficient, that is, the coefficient of t^d .

$$\begin{aligned} p_d &= 2 \cdot \sum_{j \in \mathbb{Z}} m_j \cdot 2^d \cdot p_d \\ &= 2^d \cdot p_d \end{aligned}$$

From $d > 0$ it follows that $p_d = 0$.

Thus the degree d must be zero and by normalization it must be $p_0 = 1$. We can easily check that this constant polynomial is actually refined by any mask with sum $\frac{1}{2}$.

- Case $n > 0$: Induction step

The induction hypothesis is that for any mask with coefficient sum 2^{-n} we can determine a refining polynomial of degree $n - 1$, that is unique when normalized so that the leading coefficient is 1. The induction claim is that for a mask m with sum 2^{-n-1} we have a uniquely determined polynomial of degree n with leading coefficient 1. We observe that $2 \cdot m$ satisfies the premise of the induction hypothesis and thus there is a polynomial q of degree $n - 1$ that is refined by $2 \cdot m$ and that is unique when normalized. Since the coefficient sum of $2 \cdot m$ is at most $\frac{1}{2}$, it is different from 1 and thus the first case of Lemma 2.3 applies. It lets us obtain the m -refinable polynomial p in the following way: Let Q be the antiderivative polynomial of q where the constant term is zero, then it is

$$\begin{aligned} \forall k > 0 \quad p_k &= \frac{Q_k}{Q_n} \\ p_0 &= \frac{2}{Q_n \cdot (1 - 2^{-n})} \cdot \sum_{j \in \mathbb{Z}} m_j \cdot \hat{Q}(-j) \quad . \end{aligned}$$

Now we turn to the question of why p is uniquely determined. Assume we have two normalized polynomial functions \hat{p}_0 and \hat{p}_1 that are both refined by mask m . Then their derivatives \hat{p}_0' and \hat{p}_1' are refined by mask $2 \cdot m$. Due to the induction hypothesis the normalized polynomial functions of \hat{p}_0' and \hat{p}_1' are equal. The first case of Lemma 2.3 implies that the antiderivatives with respect to \hat{p}_0' and \hat{p}_1' have the same integration constant, and thus $p_0 = p_1$. \square

Theorem 2.2. *Given a polynomial p of degree n , there is a uniquely defined mask m of support in $\{0, \dots, n\}$ that refines \hat{p} .*

For the proof of that theorem we introduce some matrices.

Definition 2.4. *We express shrinking a polynomial by factor k by the matrix S_k .*

$$\begin{aligned} S_k &\in \mathbb{R}^{\{0, \dots, n\} \times \{0, \dots, n\}} \\ S_k &= \text{diag}(1, k, \dots, k^n) \end{aligned}$$

We represent translation of a polynomial by 1 by the matrix T and translation of a distance i by the power T^i .

$$T \in \mathbb{R}^{\{0,\dots,n\} \times \{0,\dots,n\}}$$

$$(T^i)_{j,k} = \begin{cases} \binom{k}{j} \cdot (-i)^{k-j} & : j \leq k \\ 0 & : j > k \end{cases}$$

The proof of Theorem 2.2 follows.

Proof. We define the matrix P that consists of translated polynomials as columns.

$$P = (T^0 p, \dots, T^n p)$$

Now computing m is just a matter of solving the simultaneous linear equations

$$p = 2 \cdot S_2 P m \quad .$$

We only have to show that P is invertible. We demonstrate that by doing a kind of LU decomposition, that also yields an algorithm for actually computing m . Our goal is to transform P into triangular form by successive subtractions of adjacent columns. We define

$$\Delta p = T p - p$$

what satisfies

$$\Delta(T^k p) = T^k \Delta p \quad .$$

In the first step we replace all but the first columns of P by differences, yielding the matrix U_1 .

$$\begin{aligned} U_1 &= (p, \Delta p, T \Delta p, \dots, T^{n-1} \Delta p) \\ &= P \cdot L_1^{-1} \end{aligned}$$

$$L_1^{-1} = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \quad L_1 = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

In the second step we replace all but the first two columns of U_1 by differences (of the contained differences), yielding the matrix U_2 .

$$\begin{aligned} U_2 &= (p, \Delta p, \Delta^2 p, T \Delta^2 p, \dots, T^{n-2} \Delta^2 p) \\ &= U_1 \cdot L_2^{-1} \end{aligned}$$

8 *Henning Thielemann*

$$L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \quad L_2 = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

We repeat this procedure n times, until we get

$$\begin{aligned} U_n &= (p, \Delta p, \Delta^2 p, \dots, \Delta^n p) \\ P &= U_n \cdot L_n \cdot \dots \cdot L_2 \cdot L_1 \quad . \end{aligned}$$

Since the k -th difference of a polynomial of degree n is a polynomial of degree $n - k$, the matrix U_n is triangular and invertible. Thus P is invertible. We get

$$\begin{aligned} m &= \frac{1}{2} \cdot P^{-1} \cdot S_{\frac{1}{2}} p \\ &= \frac{1}{2} \cdot L_1^{-1} \cdot L_2^{-1} \cdot \dots \cdot L_n^{-1} \cdot U_n^{-1} \cdot S_{\frac{1}{2}} p \quad , \end{aligned}$$

where the product $U_n^{-1} \cdot S_{\frac{1}{2}} p$ can be computed by back-substitution and the multiplications with L_j^{-1} mean computing some differences of adjacent vector elements. \square

Theorem 2.3. *If p is a polynomial of degree n and m is a mask that refines \hat{p} , then for every mask v the mask $m + v * (1, -1)^{n+1}$ refines \hat{p} as well. Only masks of this kind refine \hat{p} . The expression $(1, -1)^{n+1}$ denotes the $(n + 1)$ -th convolution of the mask $\delta_0 - \delta_1$, that is $(1, -1)^0 = (1)$, $(1, -1)^1 = (1, -1)$, $(1, -1)^2 = (1, -2, 1)$ and so on.*

Proof. We denote the convolution of a mask m with a polynomial by the matrix C_m .

$$\begin{aligned} C_m &\in \mathbb{R}^{\{0, \dots, n\} \times \{0, \dots, n\}} \\ C_m &= \sum_{i=\nu}^{\kappa} m_i \cdot T^i \end{aligned}$$

The refinement equation can be written

$$p = 2 \cdot S_2 C_m \cdot p \quad .$$

Since C_m is a LAURENT matrix polynomial expression with respect to T , it holds

$$\begin{aligned} C_{h+g} &= C_h + C_g \\ C_{h*g} &= C_h \cdot C_g \quad . \end{aligned}$$

$$\begin{aligned}
& 2 \cdot S_2 C_{m+v*(1,-1)^{n+1}} \cdot p \\
&= S_2(2 \cdot C_m \cdot p) + S_2(2 \cdot C_{v*(1,-1)^{n+1}} \cdot p) \\
&= p + S_2(2 \cdot C_v \cdot (C_{(1,-1)}^{n+1} \cdot p)) \\
& (n+1)\text{-th difference of an } n\text{-degree polynomial vanishes: } C_{(1,-1)}^{n+1} \cdot p = 0 \\
&= p
\end{aligned}$$

We still have to show, that refining masks of \hat{p} always have the form $m + v * (1, -1)^{n+1}$. Consider a mask h_1 that refines \hat{p} . By computing the LAURENT polynomial division remainder with respect to the divisor $(1, -1)^{n+1}$ we can reduce h_1 to a mask h_0 that has support in $\{0, \dots, n\}$ and we can reduce m to a mask g with support in $\{0, \dots, n\}$, too.

$$\begin{aligned}
m &= g + v_0 * (1, -1)^{n+1} \\
h_1 &= h_0 + v_1 * (1, -1)^{n+1}
\end{aligned}$$

From the above considerations we conclude that both g and h_0 refine \hat{p} and the uniqueness property in Theorem 2.2 eventually gives us $g = h_0$, thus

$$h_1 = m + (v_1 - v_0) * (1, -1)^{n+1} \quad . \quad \square$$

Remark 2.2. By adding terms of the form $v * (1, -1)^{n+1}$ we can shift the support of a mask and still refine the same polynomial.

2.3. Example

For better comprehension of the theorems of the previous section let us examine a longer example. We would like to illustrate that the same refinement mask can refine different functions. However, if we restrict the function class to, say, continuous compactly supported functions or to polynomial functions, then a refinement mask is associated with a unique function. We would like to compare a continuous compactly supported function with a polynomial function, both being refinable with respect to the same mask. Unfortunately this is not possible, since for the former type of functions we need masks with sum 1, whereas for polynomial functions we need masks with sums that are powers of two that are smaller than 1. So, we are going to compare antiderivatives of the quadratic B-spline and polynomial functions that are refinable with respect to the masks $\frac{1}{16} \cdot (1, 3, 3, 1)$, $\frac{1}{32} \cdot (1, 3, 3, 1)$, $\frac{1}{64} \cdot (1, 3, 3, 1)$ according to Lemma 2.3.

Figure 2 shows the refinable functions. Since the quadratic B-spline as in Figure 1 is refinable with respect to the mask $\frac{1}{8} \cdot (1, 3, 3, 1)$, its antiderivative as in the top-left plot in Figure 2 is refined by $\frac{1}{16} \cdot (1, 3, 3, 1)$. The sum of this mask is $\frac{1}{2}$ and thus the only refinable polynomial for that mask is a constant polynomial. We normalize it to 1. This is shown in the top-right plot of Figure 2.

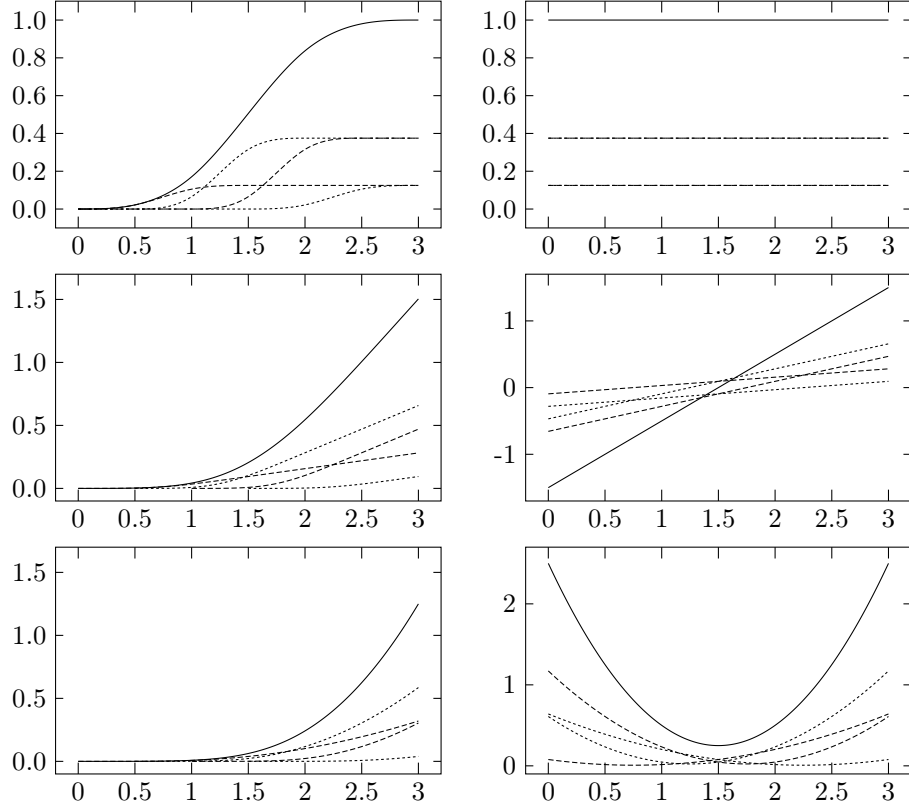


Figure 2. Refinement of antiderivatives of the quadratic B-spline (left column) and polynomial functions (right column) with respect to the masks $\frac{1}{16} \cdot (1, 3, 3, 1)$, $\frac{1}{32} \cdot (1, 3, 3, 1)$, $\frac{1}{64} \cdot (1, 3, 3, 1)$ (first to last row). The meaning of the lines is the same as in Figure 1.

The second row is associated with mask $\frac{1}{32} \cdot (1, 3, 3, 1)$. We get the refined polynomial by integrating the function $t \mapsto 1$ as in the proof of Theorem 2.1:

$$\begin{aligned}\widehat{Q}(t) &= t \\ p_1 &= 1 \quad p_0 = \frac{2}{1/2} \cdot \frac{1}{32} \cdot (1 \cdot 0 + 3 \cdot (-1) + 3 \cdot (-2) + 1 \cdot (-3)) = -\frac{3}{2} \\ \widehat{p}(t) &= -\frac{3}{2} + t\end{aligned}$$

That is, $t \mapsto -\frac{3}{2} + t$ is refined by $\frac{1}{32} \cdot (1, 3, 3, 1)$. We repeat this procedure in order

to get to the last row of Figure 2. We start with the antiderivative of $t \mapsto -\frac{3}{2} + t$:

$$\begin{aligned}\widehat{Q}(t) &= -\frac{3 \cdot t}{2} + \frac{t^2}{2} \\ p_2 &= 1 \quad p_1 = -3 \quad p_0 = \frac{2}{1/2 \cdot 3/4} \cdot \frac{1}{64} \cdot (1 \cdot 0 + 3 \cdot 2 + 3 \cdot 5 + 1 \cdot 9) = \frac{5}{2} \\ \widehat{p}(t) &= \frac{5}{2} - 3 \cdot t + t^2\end{aligned}$$

We want to check whether the obtained polynomial function $t \mapsto \frac{5}{2} - 3 \cdot t + t^2$ is actually refined by $\frac{1}{64} \cdot (1, 3, 3, 1)$ according to (2.1):

$$\begin{array}{ll}\widehat{p}(2t-0) = \frac{5}{2} - 6 \cdot t + 4 \cdot t^2 & 1 \cdot \widehat{p}(2t-0) = \frac{5}{2} - 6 \cdot t + 4 \cdot t^2 \\ \widehat{p}(2t-1) = \frac{13}{2} - 10 \cdot t + 4 \cdot t^2 & 3 \cdot \widehat{p}(2t-1) = \frac{39}{2} - 30 \cdot t + 12 \cdot t^2 \\ \widehat{p}(2t-2) = \frac{25}{2} - 14 \cdot t + 4 \cdot t^2 & 3 \cdot \widehat{p}(2t-2) = \frac{75}{2} - 42 \cdot t + 12 \cdot t^2 \\ \widehat{p}(2t-3) = \frac{41}{2} - 18 \cdot t + 4 \cdot t^2 & 1 \cdot \widehat{p}(2t-3) = \frac{41}{2} - 18 \cdot t + 4 \cdot t^2 \\ \hline & 80 - 96 \cdot t + 32 \cdot t^2\end{array}$$

$$\frac{2}{64} \cdot (80 - 96 \cdot t + 32 \cdot t^2) = \frac{5}{2} - 3 \cdot t + t^2$$

That is, the mask actually refines the polynomial function.

In the next step of our example we want to determine refinement masks for our polynomial functions by means of Theorem 2.2. However, it is already clear that we will not get the mask $\frac{1}{64} \cdot (1, 3, 3, 1)$ as a result because Theorem 2.2 only promises a mask of size 3 for a quadratic polynomial. Nonetheless this smaller mask should be compatible to the original one in the sense of Theorem 2.3.

$$\begin{aligned}\widehat{p}(t) &= \frac{5}{2} - 3 \cdot t + t^2 \\ \widehat{\Delta p}(t) &= \widehat{p}(t-1) - \widehat{p}(t) \\ &= \left(\frac{5}{2} - 3 \cdot (t-1) + (t-1)^2 \right) - \left(\frac{5}{2} - 3 \cdot t + t^2 \right) \\ &= 4 - 2 \cdot t \\ \widehat{\Delta^2 p}(t) &= \widehat{\Delta p}(t-1) - \widehat{\Delta p}(t) \\ &= (4 - 2 \cdot (t-1)) - (4 - 2 \cdot t) \\ &= 2\end{aligned}$$

12 *Henning Thielemann*

$$\begin{aligned}
U_2 = (p, \Delta p, \Delta^2 p) &= \begin{pmatrix} \frac{5}{2} & 4 & 2 \\ -3 & -2 & 0 \\ 1 & 0 & 0 \end{pmatrix} \\
m = \frac{1}{2} \cdot L_1^{-1} \cdot L_2^{-1} \cdot U_2^{-1} \cdot S_{\frac{1}{2}} p &= L_1^{-1} \cdot L_2^{-1} \cdot U_2^{-1} \cdot \frac{1}{8} \cdot \begin{pmatrix} 10 \\ -6 \\ 1 \end{pmatrix} \\
&= L_1^{-1} \cdot L_2^{-1} \cdot \frac{1}{32} \cdot \begin{pmatrix} 4 \\ 6 \\ 3 \end{pmatrix} = L_1^{-1} \cdot \frac{1}{32} \cdot \begin{pmatrix} 4 \\ 3 \\ 3 \end{pmatrix} \\
&= \frac{1}{32} \cdot \begin{pmatrix} 1 \\ 0 \\ 3 \end{pmatrix}
\end{aligned}$$

We leave it to the reader to verify that this mask actually refines our polynomial.

The last thing we want to check is, that the difference between the short mask and the original mask is a convolutional multiple of $(1, -1)^3$ as stated by Theorem 2.3:

$$\frac{1}{64} \cdot (1, 3, 3, 1) - \frac{1}{32} \cdot (1, 0, 3, 0) = \frac{1}{64} \cdot (-1, 3, -3, 1) = -\frac{1}{64} \cdot (1, -1)^3 \quad .$$

2.4. The cascade algorithm

We want to close the section on the theoretical results with an alternative way to compute the polynomial that is refined by a mask. It is an iterative algorithm for the approximate computation of a polynomial, and it is analogous to the *cascade algorithm* known for refinable functions of bounded support.⁵ The refinement relation

$$p = 2 \cdot S_2 C_m \cdot p$$

is interpreted as a recursively defined function sequence with

$$p_{j+1} = 2 \cdot S_2 C_m \cdot p_j \quad .$$

This iteration is in fact the vector iteration method for computing the eigenvector that corresponds to the largest eigenvalue.

Theorem 2.4. *Given a mask m that sums up to 2^{-n-1} for a given natural number n and a starting polynomial p_0 of degree n that is not orthogonal to the refined polynomial, the recursion*

$$p_{j+1} = 2 \cdot S_2 C_m \cdot p_j$$

converges and the limit polynomial $\lim_{j \rightarrow \infty} p_j$ is refined by m .

An appropriate choice for p_0 is $(0, \dots, 0, 1)$.

Proof. The matrix S_2C_m expands to

$$S_2C_m = \begin{cases} 2^j \cdot \binom{k}{j} \cdot \sum_{i=\nu}^{\kappa} (-i)^{k-j} \cdot m_i & : j \leq k \\ 0 & : j > k \end{cases}$$

and thus is of upper triangular shape. This implies that the diagonal elements $2^j \cdot \sum_{i=\nu}^{\kappa} m_i$ for $j \in \{0, \dots, n\}$ are the eigenvalues. Because the mask sums up to 2^{-n-1} the eigenvalues of $2 \cdot S_2C_m$ are $\{1, \dots, 2^{-n}\}$. That is, the largest eigenvalue is 1 and it is isolated. These are the conditions for the vector iteration method, consequently the iteration converges to a vector that is a fixed point of the refinement operation. \square

Remark 2.3. The eigenvectors of the eigenvalues are the respective derivatives of the main refinable polynomial.

3. Implementation

The presented conversions from masks to polynomials and back are implemented in the functional programming language Haskell as module `MathObj.RefinementMask2` of the NumericPrelude project⁹. However, note that the definition of the Haskell functions slightly differ from this paper, since the factor 2 must be part of the mask.

4. Related work

So far, refinable functions were mostly explored in the context of wavelet theory. In this context an important problem was to design refinement masks that lead to smooth finitely supported refinable functions.^{10,11} It was shown that smoothness can be estimated the following way: Decompose the refinement mask m into the form $(1, 1)^n * v$, where n is chosen maximally. According to Lemma 2.1 this corresponds to a convolution of functions. However, strictly speaking, v corresponds to a distribution. The exponent n represents the order of a B-spline and is responsible for the smoothness of the refinable function, whereas for v there is an eigenvalue problem, where the largest eigenvalue determines how much the smoothness of the B-spline is reduced.

The cascade algorithm⁵ was developed in order to compute numerical approximations to refinable functions. A combination of the cascade algorithm and Lemma 2.1 was used by 1 for computing scalar products and other integrals of products of refinable functions. This is required for solving partial differential equations using a wavelet ГАЛЁРКИН approach.

Discrete wavelet functions in a multiresolution analysis are defined in terms of refinable functions, but were not considered refinable functions at first. However in 7 it is shown, that wavelets are refinable with respect to infinite masks. The trick is to use polynomial division for dividing the wavelet masks of adjacent scales: If φ is refinable with respect to mask h , and ψ is a wavelet with respect to mask g and

generator φ , then ψ is refinable with respect to $\frac{g \uparrow 2}{g} \cdot h$, where $g \uparrow 2$ is g upsampled by a factor of 2.

Although polynomial functions are not in the main focus of the research on refinable functions, we got to know several discoveries of this relation after uploading our work to arXiv. The thesis 4 is the first reference known to us that explains how to obtain a polynomial function that is refined by a mask. The approach in this thesis is also based on polynomial differentiation.

The first source known to us that describes the opposite way, i.e. how to find a refining mask for a polynomial function, is 3.^a In terms of our matrices the author of that article does not just use the matrix P of shifted polynomials, but its factorization $P = A \cdot V$. The matrix V is a VANDERMONDE matrix with

$$V \in \mathbb{R}^{\{0, \dots, n\} \times \{0, \dots, n\}}$$

$$V_{i,j} = j^i$$

The matrix A is defined using the derivatives of p by $A = \left(\frac{p}{0!}, \frac{p'}{1!}, \dots, \frac{p^{(n)}}{n!} \right)$, or element-wise by

$$A \in \mathbb{R}^{\{0, \dots, n\} \times \{0, \dots, n\}}$$

$$A_{i,j} = \begin{cases} \binom{i+j}{i} \cdot p_{i+j} & : i+j \leq n \\ 0 & : i+j > n \end{cases}.$$

The TAYLOR expansion of p allows to express translations of p in terms of its derivatives:

$$\widehat{p}(t+x) = \frac{\widehat{p}(t)}{0!} + x \cdot \frac{\widehat{p}'(t)}{1!} + \dots + x^n \cdot \frac{\widehat{p}^{(n)}(t)}{n!}.$$

Thus multiplying A and V yields the sequence of shifted polynomials with $x \in \{0, -1, \dots, -n\}$.

With this factorization the invertibility of P follows obviously from the invertibility of the triangular matrix A and the invertibility of the VANDERMONDE matrix V with respect to pairwise distinct nodes.

Actually, in the article the general VANDERMONDE matrix with pairwise distinct nodes $\{\ell_0, \dots, \ell_n\}$ is used:

$$V = \begin{pmatrix} 1 & 1 & \dots & 1 \\ -\ell_0 & -\ell_1 & \dots & -\ell_n \\ \vdots & \vdots & \ddots & \vdots \\ (-\ell_0)^n & (-\ell_1)^n & \dots & (-\ell_n)^n \end{pmatrix}$$

That is, a refinement mask for a polynomial can also be found when only a certain set of $n+1$ nodes is allowed to be non-zero. The nodes may even be non-integral.

^aIt is written in 2011, but was already presented in talks in 2003.

In retrospect we could have conducted our proof of Theorem 2.2 with arbitrary nodes, too. We would have to define $P = (T^{\ell_0} \cdot p, \dots, T^{\ell_n} \cdot p)$ and then use divided differences instead of simple differences for the LU decomposition.

The paper 2 extends the previous one by an exploration of the refinability of rational functions. They find that a rational function φ is refinable if and only if there is a real sequence s ($s \in \ell_0(\mathbb{Z})$, i.e. a LAURENT polynomial) and a positive natural number k such that

$$\varphi(t) = \sum_{i \in \mathbb{Z}} \frac{s_i}{(t-i)^k} \quad \wedge \quad s|(s \uparrow 2),$$

that is, $\exists \hat{m}(z) \in \mathbb{R}[z, z^{-1}] \hat{m}(z) \cdot \hat{s}(z) = \hat{s}(z^2)$, where $2^{k-1} \cdot m$ is the refinement mask.

5. Future work

There are some obvious generalizations to be explored: refinement with respect to factors different from 2, separable multidimensional refinement and most general multidimensional refinement with respect to arbitrary dilation matrices.

Another interesting question is the following one: By Lemma 2.1 we know, that convolution of functions maps to convolution of their refinement masks. We can use this for defining a kind of convolution. In order to convolve two functions φ_0 and φ_1 , we compute refining masks m_0 and m_1 , respectively, convolve the masks and then find a function that is refined by $m_0 * m_1$. In case of polynomial functions there is no notion of convolution because the involved integrals diverge. We can however define a convolution based on refinement. Unfortunately, the mapping from a polynomial function to a refinement mask is not unique, consequently the defined convolution is not unique as well – not to speak of the arbitrary constant factor. If we choose arbitrary masks from the admissible ones, then the convolution is not distributive with addition, i.e. $\psi * (\varphi_0 + \varphi_1) = \psi * \varphi_0 + \psi * \varphi_1$ is not generally satisfied. The open question is, whether it is possible to choose masks for polynomials, such that the polynomial convolution via refinement is commutative, associative and distributive.

6. Acknowledgment

I like to thank David Larson and David Malone for pointing me to related research and Emily King for careful proof-reading and discussion of alternative proofs.

Bibliography

1. Johan M. De Villiers, Charles A. Micchelli, and Tomas Sauer. Building refinable functions from their values at integers. *Calcolo*, 37:139–158, 2000.
2. Paul Gustafson, Nathan Savir, and Ely Spears. A characterization of refinable rational functions. *American Journal of Undergraduate Research*, 5(3):11–20, November 2006.
3. Emily Jeanette King. A matricial algorithm for polynomial refinement. <http://arxiv.org/abs/1110.6061>, October 2011.

4. David Malone. *Solutions to Dilation Equations*. PhD thesis, University of Dublin, 2000.
5. Gilbert Strang. Eigenvalues of $(\downarrow 2)H$ and convergence of the cascade algorithm. *IEEE Transactions on Signal Processing*, 44:233–238, 1996.
6. Gilbert Strang and Truong Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1997.
7. Gilbert Strang, Vasily Strela, and Ding-Xuan Zhou. Compactly supported refinable functions with infinite masks. In L. Baggett and D. Larson, editors, *The Functional and Harmonic Analysis of Wavelets and Frames*, volume 247 of *Contemporary Mathematics*, pages 285–296. American Mathematical Society, 1999.
8. Henning Thielemann. *Optimally matched wavelets*. PhD thesis, Universität Bremen, March 2006.
9. Dylan Thurston, Henning Thielemann, and Mikael Johansson. numeric-prelude: An experimental alternative hierarchy of numeric type classes. <http://hackage.haskell.org/package/numeric-prelude-0.2>, September 2010.
10. Lars F. Villemoes. Sobolev regularity of wavelets and stability of iterated filter banks. *Progress in wavelet analysis and applications*, pages 243–251, 1993.
11. Lars F. Villemoes. Wavelet analysis of refinement equations. *SIAM Journal on Mathematical Analysis*, 25(5):1433–1460, 1994.